REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 6, 1995	Final 15 Apr 91 - 14 Apr 94

4. TITLE AND SUBTITLE

Research in Graph Algorithms and Combinatorial Optimization

5. FUNDING NUMBERS

6. AUTHOR(S)

Serge Plotkin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Computer Science Department
Stanford University
Stanford, CA 94305-2140

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARO 28143.6-MA

11. SUPPLEMENTARY NOTES

The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This project focused on designing fast algorithms for basic *combinatorial optimization* problems, including maximum flow, matching, multicommodity flow, and generalized flow. Many important applications are naturally stated as variants of these problems, and hence improved algorithms for these problems immediately lead to improved algorithms for a wide variety of applications.

Our goal was to improve both sequential and parallel complexity. In many applications, solving a multicommodity or a generalized flow problem is only a first step in approximately solving an NP-complete problem; in the majority of such cases there is no need to have an exact solution of the problem. One of the focuses of the project was design of efficient *approximation* algorithms.

14. SUBJECT TERMS			15. NUMBER OF PAGES: 8
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT: UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT: UNCLASSIFIED	20. LIMITATION OF ABSTRACT: UL

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z39-18
298-102

Final Report

ARO NUMBER:	28143-MA
PERIOD COVERED BY REPORT:	April 15, 1991 – April 15, 1994
GRANT NUMBER:	DAAL03-91-G-0102
NAME OF THE INSTITUTION:	Stanford University
PROJECT TITLE:	Research in Graph Algorithms and Combinatorial Optimization
PI:	Serge Plotkin Dept. of Computer Science, Stanford University (415)-723-0540 plotkin@cs.stanford.edu
STUDENTS SUPPORTED:	Orli Waarts, Anil Kamath, Omri Palmon.

1 Overview

This project focused on designing fast algorithms for basic *combinatorial optimization* problems, including maximum flow, matching, multicommodity flow, and generalized flow. Many important applications are naturally stated as variants of these problems, and hence improved algorithms for these problems immediately lead to improved algorithms for a wide variety of applications.

Our goal was to improve both sequential and parallel complexity. In many applications, solving a multicommodity or a generalized flow problem is only a first step in approximately solving an NP-complete problem; in the majority of such cases there is no need to have an exact solution of the problem. One of the focuses of the project was design of efficient *approximation* algorithms.

Substantial progress was made on designing new exact and approximation algorithms for the above mentioned problems. In particular, a new general approach was developed for solving a class of "packing" and "covering" problems. These problems can be viewed as generalizations of multicommodity flow, and include many diverse applications ranging from machine scheduling to VLSI wire routing.

Additional results include strongly polynomial algorithms (*i.e.* algorithms which running time does not depend on the precision of the input data) for several optimization problems, and a new parallel deterministic algorithms for bipartite matching. We have also developed new primitives that can be efficiently implemented and that lead to improved solutions for several basic coordination problems that one encounters when designing algorithms for shared memory multiprocessors.

2 Summary of Research Findings

2.1 Multicommodity Flow

The multicommodity flow problem is a natural generalization of the maximum-flow problem where, instead of a single commodity, we have several commodities, defined by demand/supply vectors, that have to be shipped through the network such that the total amount of flow through each edge is below its capacity. Instead of trying to satisfy all the demands, one can try to satisfy the maximum percentage of each demand. This modification is called the *concurrent flow* problem. Many optimization problems can be stated as special cases of multicommodity flow or concurrent flow. Applications include VLSI layout, network routing, and efficient simulations of one interconnection network by another.

In many cases multicommodity flow algorithm is used as a subroutine to find approximate solutions to NP-complete integer problems, and one does not need to obtain an exact solution. Until now it was believed that multicommodity flow, even with a small number of commodities, is much more complicated compared to the single commodity flow. In [18] we have considered the special case when all the edges have the same capacity, and developed an algorithm for approximating concurrent flow problem for this type of networks. In [21] we have have considered the general case of unrestricted capacities and have shown that it is possible to approximately compute a k-commodity

flow in time that is close to the time it takes to approximately compute k single-commodity flows. Our algorithm is simple to implement and it is much faster than the previously known algorithms. In addition, the fact that it consists of repetitive computations of minimum-cost flow, means that it might be more suitable for implementation in practice. (Minimum-cost flow problems are efficiently handled by the network simplex algorithm in practice.) The full version of [21] was invited to Journal of Computer and System Sciences (JCSS).

2.2 Packing and Covering problems

In [20] we have generalized results on the multicommodity flow to apply to a large class of linear programs, the so called "packing" and "covering" programs. For example, one can view job scheduling as packing jobs into machines, where the "size" of each job is its execution time, and the "capacity" is the maximum amount of execution time we allow to be assigned to a machine. The only previously known algorithms for solving these problems are based on general linear programming techniques. Our techniques greatly outperform the general methods in many applications. Our algorithm can be viewed as a Lagrangean relaxation; an important aspect of our results is that we obtain a theoretical analysis of the running time of a Lagrangean relaxation-based algorithm. The new approach yields several orders of magnitude of improvement over the best previously known running times for the scheduling of unrelated parallel machines in both the preemptive and the non-preemptive models, for the job shop problem, for the cutting-stock problem, and for the minimum-cost multicommodity flow problem. The full version of [20] was invited to Journal of Computer and System Sciences (JCSS).

2.3 Min-Cut / Max-Flow Relationships for Multicommodity Flow

In order to prove that a given multicommodity flow problem is infeasible, it is sufficient to exhibit a cut whose capacity is below the sum of the demands that are separated by the cut. The min-cut max-flow theorem for the single-commodity flow problem states that the non-existence of such a "bad" cut proves that a feasible flow does exist. This theorem, discovered in the fifties, is the basis of currently fastest single-commodity flow algorithms.

For multicommodity flow the situation is more complicated. A multicommodity flow problem can be infeasible even if the "cut condition" is satisfied. A natural question to ask is how large a "safety margin" do we need, *i.e.* how large should be the minimum ratio (over all cuts) of the capacity of the cut to the sum of the demands that are separated by this cut, in order to ensure existence of a feasible flow. A related problem is to consider a multicommodity flow problem, and to search either for a feasible flow, or for a cut whose ratio is below the above mentioned safety margin. This leads to an algorithm that finds an *approximately minimum-cut.* Approximately minimum-cut computation is a basic step for construction of approximation algorithms for a variety of NP-complete problems.

The best bound on the minimum-cut maximum-flow ratio for general multicommodity flow problems was $O(\log n \log D)$, where D is the sum of all the demands and n is the number of nodes in the graph. Note that $\log D$ can be as large as n, making this bound useless. In [11] we have improved this bound to $O(\log^2 k)$, where k is the number of commodities, proving for the first time

that the bound does not depend on the precision of the input data. In [12] we proved that if the network is sufficiently sparse, then the bound changes to $O(\log k)$. Moreover, for planar graphs when the demands are uniform (unit demand between each pair of nodes) we have improved the bound to $O(1)$.

2.4 Strongly Polynomial Algorithms

An important open problem is whether or not there exists a strongly polynomial algorithm for linear programming, *i.e.* an algorithm with running time that depends only on the number of inequalities and variables, and not on the size of the numbers involved. In recent years there has been substantial progress in this direction, and currently there are several special cases of linear programs for which strongly polynomial algorithms are known. In a joint work with Norton and Tardos [22], we have developed a technique that extends the class of linear programs solvable in strongly-polynomial algorithms. Informally, we prove that if we have a linear problem that is solvable in strongly-polynomial time, than any problem that is obtained from it by adding a constant number of additional variables or rows, is solvable in strongly-polynomial time as well. In particular, our technique leads to the first strongly-polynomial algorithm for the concurrent multicommodity flow. (Concurrent multicommodity flow is similar to multicommodity flow, but instead of having to satisfy all the demands, we are required to satisfy a percentage of each demand). This paper was invited to Journal of Algorithms.

2.5 Large scale optimization

Kamath has been working on design and implementation of algorithms to compute bounds on solutions to NP-hard problems. A new technique to design such algorithms (based on the interior-point method for linear programming) is presented in [8]. This technique has been applied towards computing bounds and approximate solutions for graph partitioning, coloring, independent set, and maximum satisfiability.

Minimum-cost assignment in bipartite graphs is a basic problem that is encountered in applications ranging from scheduling to vision. Kamath has been working on implementing a variant of dual-projective interior-point linear programming method especially geared towards solving very large assignment problems. The algorithm and implementation results are presented in [7]. Object-oriented implementation of an interior-point algorithm to solve large multicommodity flow problems is presented in [10]. These problems, which arise in a wide range of applications (including network design, crew scheduling, bandwidth allocation, etc.) have a special structure that can be effectively exploited in an object-oriented setup. The resulting implementation not only reduced the space requirement (often a problem with interior-point implementations) but also improved the speed.

2.6 Graph separators

A node separator of a graph is a set of nodes whose removal separates the graph into two roughly equal components. Finding small separators has numerous applications including VLSI layout and

simulation of one interconnection network by another. In general, the fact that a certain family of graphs has small separators can be used to construct a variety of divide and conquer algorithms for these graphs.

One natural way of defining graph families is based on the notion of a *graph minor.* We say that graph H is a minor of G if H can be obtained from G by contracting some of G's edges and deleting some other edges and nodes. For example, Kuratowski theorem states that a graph is planar if and only if it does not include a clique of 5 nodes (K_5) or a complete bipartite graph on 6 nodes ($K_{3,3}$) as minors.

In [4] we prove that the fact that a graph does not have K_h as a minor implies that it has an $O(h\sqrt{n \log n})$-size separator. This is an improvement of previously best known bound for the case when $h = \omega(\log n)$. We also show how to apply these techniques to simulate a large class of graphs on a hypercube and to design of efficient out-of-core relaxation schemes.

2.7 Algorithms for Asynchronous Shared-Memory Multiprocessor

A snapshot scan algorithm takes an "instantaneous" picture of a region of shared memory that may be updated by concurrent processes. Many complex and difficult shared memory algorithms can be greatly simplified by structuring them around the snapshot scan abstraction. Unfortunately, the substantial decrease in conceptual complexity is quite often counterbalanced by an increase in computational complexity. In [24] we introduce the notion of a weak snapshot scan, a slightly weaker primitive that has a more efficient implementation. We propose the following methodology for using this abstraction: First, design and verify an algorithm using the more powerful snapshot scan, and second, replace the more powerful but less efficient snapshot with the weaker but more efficient snapshot, and show that the weaker abstraction nevertheless suffices to ensure the correctness of the enclosing algorithm.

We give two examples of algorithms whose performance can be enhanced while retaining a simple modular structure: randomized consensus and bounded concurrent timestamping. The resulting randomized consensus protocol is the fastest known protocol that uses only bounded values. Our technique also allow us to simplify and improve the complexity of the timestamping protocol.

2.8 Contention in Shared-Memory Multiprocessor

Inability of standard models of shared memory multiprocessors to take memory contention into account is one of their main shortcomings. Two algorithms might be indistinguishable in these models even if one causes many concurrent accesses to the same memory cell, while the other makes sure such accesses never happen. In many of the current multiprocessor architectures concurrent access causes significant degradation in performance, and should be avoided as much as possible.

Together with collaborators, Waarts has proposed a new model that facilitates comparison of algorithms with respect to contention. Using this model, she has provided the first formal explanation of the experimental results indicating superiority of "counting networks" over conventional approaches for implementation of shared counters.

2.9 On-line Optimization

Support for *virtual circuits* is one of the basic services provided by both existing and future high-speed communication networks. In order to use the network (say, transmit video signal from one point to another) the user requests a (virtual) circuit to be established between these points. The network has to choose a path between the endpoints of the circuit and allocate sufficient bandwidth along this path. Because of hardware limitations, rerouting of circuits, i.e. moving a circuit from one path to another, is either forbidden or heavily discouraged.

It is easy to see that bad routing decisions may lead to very poor utilization of the total available bandwidth. One of the main thrusts of this research project was development of novel online strategies for virtual circuit routing that lead to *provably efficient* bandwidth utilization. The strategies are based on the recent combinatorial approximation algorithms for multicommodity flow [18, 20].

In [6] we consider online routing of permanent virtual circuits (PVCs) and describe an algorithm that achieves an $O(\log n)$ competitive factor with respect to congestion, where n is the number of nodes in the network. Roughly speaking, this means that the congestion on the maximum congested link achieved by this algorithm will never exceed $O(\log n)$ times the maximum congestion achieved by the best possible omniscient algorithm that knows all of the future requests. In [6] we extended these results to the switched virtual circuit routing case (SVC), *i.e.* the case where each circuit has an associated duration. The competitive ratio of the resulting routing strategy is bounded by $O(\log nT)$, where T is the maximum duration of a circuit.

Instead of minimizing congestion, sometimes it is more appropriate to maximize the throughput, i.e. the average number of transmitted bits per unit of time. In [5], we present an $O(\log nT)$-competitive algorithm with respect to throughput. We also show that the performance achieved by this algorithm is asymptotically optimal. A variant of this algorithm (as applied to permanent virtual circuit routing) was implemented in Bell Laboratories and will be used in the ATM switches currently manufactured by AT&T. Together with Kamath, we have implemented a switched virtual circuit variant and are currently testing it on data supplied by AT&T.

Publications

[1] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, January 1994.

[2] M. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, É. Tardos, and D. Williamson. Improved approximation algorithms for network design problems. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[3] T. Feder, N. Megiddo, and S. Plotkin. A sublinear parallel algorithm for stable matching. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, January 1994.

[4] S. Plotkin, S. Rao, and W. Smith. Sublinear separators for graphs with large forbidden minors. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, January 1994.

[5] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive on-line routing. In *Proc. 34th IEEE Annual Symposium on Foundations of Computer Science*, November 1993.

[6] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. Pruhs, and O. Waarts. On-line load balancing of temporary tasks. In *Proc. Workshop on Algorithms and Data Structures*, pages 119–130, August 1993.

[7] K. G. Ramakrishnan, N. Karmarkar, and A. Kamath. An Approximate Dual Projective Algorithm for Solving Assignment Problems. In *DIMACS Series in Discrete Math and Theor. Comp. Sci.*, Vol. 12, pages 431–452, 1993.

[8] A. Kamath and N. Karmarkar. An $O(nL)$ Iteration Algorithm for Computing Bounds in Quadratic Optimization Problems. In *Complexity in Numerical Optimization*, World Scientific, pages 254–268, 1993.

[9] A. Kamath, N. Karmarkar, K. G. Ramakrishnan, M. G.C. Resende. An Interior Point approach to Boolean Vector Function Synthesis. *Mid-West Conference on Circuits and Systems*, 1993.

[10] A. Kamath, N. Karmarkar, K. G. Ramakrishnan. Computational and Complexity results for an Interior Point Algorithm on MultiCommodity Flow Problems. *Networks*, 1993 .

[11] S. Plotkin and É. Tardos. Improved bounds on the max-flow min-cut ratio for multicommodity flows. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, May 1993.

[12] P. Klein, S. Plotkin, and S. Rao. Planar graphs, multicommodity flow, and network decomposition. To appear in *Proc. 25th ACM Symposium on the Theory of Computing*, May 1993.

[13] C. Dwork, M. P. Herlihy, and O. Waarts. Contention in shared memory algorithms. *Proc. 25th ACM Symposium on the Theory of Computing*, May 1993.

[14] A. Goldberg, B. Maggs, and S. Plotkin. A parallel algorithm for reconfiguring a multibutterfly network with faulty switches. *IEEE Trans. on Computers*, 1993.

[15] A. Goldberg and S. Plotkin. Lecture notes: Topics in Combinatorial Optimization. Technical Report STAN-CS-92-1447, Department of Computer Science, Stanford University, November 1992.

[16] J. Aspnes and O. Waarts. Randomized Consensus in expected $O(n \log^2 n)$ operations per processor. In *Proc. 33rd Symp. Foundations of Computer Science*, pp. 137-146, October 1992.

[17] T. Fischer, A. Goldberg, and S. Plotkin. Approximating matching in parallel. *Information Processing Let.*, 1993.

[18] P. Klein, S. Plotkin, C. Stein, and É. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, June 1994.

[19] J. Orlin, S. Plotkin, and É. Tardos. Polynomial Dual Network Simplex. *Mathematical Programming*, 60:255–276, 1993.

[20] S. Plotkin, D. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math of Oper. Research*, 1994. Accepted for publication.

[21] T. Leighton, F. Makedon, S. Plotkin, C. Stein, S. Tragoudas, and É. Tardos. Fast approximation algorithms for multicommodity flow problem. *J. Comp. and Syst. Sci.*, 1992.

[22] C. Haibt-Norton, S. Plotkin, and É. Tardos. Using Separation Algorithms in Fixed Dimension. *Journal of Algorithms*, 13:79–98, 1992.

[23] A. Goldberg, S. Plotkin, and P. Vaidya. Sublinear-Time Parallel Algorithms for Matching and Related Problems. *Journal of Algorithms*, 14(2):180–213, March 1993.

[24] C. Dwork, M. P. Herlihy, S. Plotkin, and O. Waarts. Time-lapse snapshots. In *Proc. 1st Israeli Symposium on Theoretical Computer Science*, May 1992.